## MCA509 COMPILER CONSTRUCTION

| L | T | P | Cr |
|---|---|---|---|
| 3 | 0 | 2 | 4.0 |

**Course Objective:** Gain the working knowledge of the major phases of compilation. Develop the ability to use formal attributed grammars for specifying the syntax and semantics of programming languages. Learn about function and complexities of modern compilers and design a significant portion of a compiler.

**Introduction to compiling:** Compilers, Analysis of the source program, the phases of Compiler, Compilation and Interpretation, Bootstrapping and Cross compiler.

**Lexical Analysis:** Need of Lexical analyzer, Tokens and regular expressions, Generation of lexical analyzer from DFA, Introduction to LEX and program writing in LEX.

**Syntax Analysis:** Need for syntax analysis and its scope, Context free grammar, Top down parsing, bottom up parsing, backtracking and their automatic generation, LL(1) Parser, LR Parser, LR(0) items, SLR(1), LALR(1), Canonical Parsing, Introduction to YACC and Integration with LEX.

**Error Analysis:** Introduction to error analysis, detection, reporting and recovery from compilation errors, Classification of error-lexical, syntactic and semantic with examples, Detection of syntactic error in LL and LR parsers, panic mode error recovery and error recovery in YACC tool.

**Static semantics and Intermediate Code generation:** Need for various static semantic analyses in declaration processing, name and scope analysis, S-attribute def. and their evaluation in different parsing, Semantic analysis through S-attribute grammar, L-attribute def. and their evaluation.

**Run time Environment:** Need for runtime memory management, Address resolution of runtime objects at compile time, Type checking, Language features influencing run time memory management, Parameter passing mechanism, Division of memory into code, stack, heap and static, Activation record, Dynamic memory management, garbage collection.

**Code Generation:** Code generation for expressions, Issues in efficient code generation, Sethi Ullman algorithm, Dynamic programming approach for optimal code generation tree, Introduction to retarget able code generation, code generation for control structures.

**Code Optimization:** Need for code optimizations, Local and global optimization, Control flow analysis, Data flow analysis, performing global optimizations, Graph colouring in optimization, Live ranges of run time values.

**Laboratory work:** To implementing concepts of various phases of a compiler in a high level language. To implement a mini working compiler.

**Text Books:**

1. Aho A.V., Ullman J. D., Sethi R., Compilers Principles, Techniques and Tools, Pearson Education, Edition 2nd ,2005
2. John Levine, Tony Mason, Doug Brown, Lex and Yacc, O'Reilly, Edition 2nd1992
3. Kenneth C. Louden, Compiler Construction and Practices, Thomson Publication, Edition 2nd, 1997.
4. Dhamdhere, Compiler Construction. Macmillan Publication Edition 2nd , 2008